Package 'lrgpr'

January 7, 2014

Title lrgpr

Version 0.0.1

Date December 20, 2013

Author Gabriel E. Hoffman

Maintainer Gabriel E. Hoffman <gabriel.hoffman@mssm.edu>

Description Fit a Low Rank Gaussian Process Regression (LRGPR) / Linear Mixed Model (LMM) for large datasets. These models are widely used in statistical genetics as a test of association while correcting for the confounding effects of kinship and population structure.

Depends R (>= 3.0.0), methods, Rcpp, RcppGSL, RcppProgress, MASS,parallel, doParallel, formula.tools, BH, bigmemory (>= 4.4.7),aod

LinkingTo Rcpp, RcppGSL

URL

License GPL (>= 2)

Collate 'genericFunctions.R' 'lrgpr.R' 'plink.R'

R topics documented:

`AIC.lrgpr` *Akaike's Information Criterion (AIC)*

Description

AIC for model fit by `lrgpr`

Usage

```
AIC.lrgpr(object, ..., k = 2)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments
`k`	for compotability, not used

BIC.lrgpr *Bayesian Information Criterion (BIC)*

Description

BIC for model fit by `lrgpr`

Usage

```
BIC.lrgpr(object, ...)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments

coefficients.lrgpr *Extract Model Coefficients*

Description

Coefficients estimated with `lrgpr`

Usage

```
coefficients.lrgpr(object)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments

`convertToBinary` *Convert ASCII to binary file*

Description

'convertToBinary' converts TPED/DOSAGE/GEN files to binary format

Usage

```
convertToBinary(filename, filenameOut, format)
```

Arguments

`filename`	file to be converted
`filenameOut`	name of binary file produced
`format`	specify 'TPED', 'DOSAGE' or 'GEN'

Details

- TPED: plink file can be in either –recode or –recode12 format
- DOSAGE: file follows plink format: http://pngu.mgh.harvard.edu/~purcell/plink/dosage.shtml

Example:

SNP A1 A2 F1 I1 F2 I2 F3 I3

rs0001 A C 0.98 0.02 1.00 0.00 0.00 0.01

rs0002 G A 0.00 1.00 0.00 0.00 0.99 0.01

where the F* values correspond to the dosage values

- GEN: file follow OXFORD format

`cooks.distance.lrgpr`
Regression Deletion Diagnostics

Description

Basic quantities for regression deletion diagnostics from fit of `lrgpr`

Usage

```
cooks.distance.lrgpr(model,
  infl = lm.influence(model, do.coef = FALSE),
  res = weighted.residuals(model),
  sd = sqrt(deviance(model)/df.residual(model)),
  hat = infl$hat, ...)
```

Arguments

model	model fit with lrgpr
infl	influence structure as returned by lm.influence
res	residuals
sd	standard deviation to use
hat	hat values
...	other arguments

---

criterion.lrgpr *Compute AIC/BIC/GCV for lrgpr() model as rank changes*

Description

'criterion.lrgpr' evaluate information criteria to select an optimal rank

Usage

```
criterion.lrgpr(formula, features, order,
  rank = c(seq(1, 10), seq(20, 100, by = 10), seq(200, 1000, by = 100)))
```

Arguments

formula	standard linear modeling syntax as used in 'lm'
features	matrix from which the SVD is performed
order	sorted indices of features. When rank is 10, decomp = svd(X[,order[1:10]])
rank	array with elements indicating the number of confounding covariates to be used in the random effect.

See Also

plot.criterion.lrgpr, cv.lrgpr

#'

Examples

```
n = 300
p = 5000
X = matrix(sample(0:2, n*p, replace=TRUE), nrow=n)

dcmp = svd(X)

# simulate response
h_sq = .8
eta = dcmp$u[,1:2] %*% rgamma(2, 2, 1)
error_var = (1-h_sq) / h_sq  * var(eta)
```

```
y = eta + rnorm(n, sd=sqrt(error_var))

# Get ordering based on marginal correlation
i = order(cor(y, X)^2, decreasing=TRUE)

# Fit AIC / BIC / GCV based on degrees of freedom
fit = criterion.lrgpr( y ~ 1, features=X, order=i)

plot.criterion.lrgpr(fit)
```

`cv.lrgpr` *Cross-validation for LRGPR*

Description

'cv.lrgpr' fits cross-validation for multiple ranks of the LRGPR

Usage

```
cv.lrgpr(formula, features, order, nfolds = 10,
  rank = c(seq(0, 10), seq(20, 100, by = 10), seq(200, 1000, by = 100)),
  nthreads = 1)
```

Arguments

`formula`	standard linear modeling syntax as used in 'lm'
`features`	matrix from which the SVD is performed
`order`	sorted indices of features. When rank is 10, decomp = svd(X[,order[1:10]])
`nfolds`	number of training sets
`rank`	array with elements indicating the number of confounding covariates to be used in the random effect.
`nthreads`	number of threads to be used

Examples

```
n = 300
p = 5000
X = matrix(sample(0:2, n*p, replace=TRUE), nrow=n)

dcmp = svd(X)

# simulate response
h_sq = .8
eta = dcmp$u[,1:2] %*% rgamma(2, 2, 1)
error_var = (1-h_sq) / h_sq  * var(eta)
y = eta + rnorm(n, sd=sqrt(error_var))

# Get ordering based on marginal correlation
```

```
i = order(cor(y, X)^2, decreasing=TRUE)

# Fit cross-validation
fit = cv.lrgpr( y ~ 1, features=X, order=i)

plot.cv.lrgpr(fit)
```

---

df.residual.lrgpr *Residual Degrees-of-Freedom*

Description

Residual df from fit of `lrgpr`

Usage

```
df.residual.lrgpr(object, ...)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments

---

error.bar *Plot Error Bars*

Description

Plot error bars for a confidence interval

Usage

```
error.bar(x, y, upper, lower = upper, length = 0.1, ...)
```

Arguments

`x`	x-axis position
`y`	y-axis position
`upper`	height of bar above y
`lower`	height of bar below y
`length`	horizontal length of the error bar
`...`	arguments for arrows(...)

getAlleleFreq *Calculate allele frequency*

Description

Calculate allele frequency

Usage

```
getAlleleFreq(X, nthreads = detectCores(logical = TRUE))
```

Arguments

X	matrix where each column is a marker coded 0,1,2 or with dosage values in this range
nthreads	number of threads to use

getAlleleVariance *EValuate variance for each column*

Description

EValuate variance for each column

Usage

```
getAlleleVariance(X,
  nthreads = detectCores(logical = TRUE))
```

Arguments

X	matrix where each column is a marker
nthreads	number of threads to use

getMissingCount *Count missing values*

Description

Count missing values

Usage

```
getMissingCount(X,
  nthreads = detectCores(logical = TRUE))
```

Arguments

X	matrix where each column is a marker
nthreads	number of threads to use

glmApply *Fit standard linear or logistic model for many markers*

Description

'glmApply' is analogous to 'lrgprApply', but fits standard linear or logistic models for many markers

Usage

```
glmApply(formula, features, terms = NULL,
  family = gaussian(), useMean = TRUE,
  nthreads = detectCores(logical = TRUE),
  univariateTest = TRUE, multivariateTest = FALSE,
  verbose = FALSE)
```

Arguments

formula	standard linear modeling syntax as used in 'lm'. SNP is a place holder for the each successive column of features
features	a matrix where the statistical model is evaluated with SNP if formula replace by each column successively
terms	indices of the coefficients to be tested. The indices corresponding to SNP are used if terms is not specified
family	gaussian() for a continuous response, and binomial() to fit a logit model for a binary response

useMean — if TRUE, replace missing entries with column mean. Otherwise, do not evaluate the model for that column

nthreads — number of to use for parallel execution

univariateTest — perform univariate hypothesis test for each response for each feature in the loop variable

multivariateTest — perform multivariate hypothesis test for each response (if more than one) for each feature. Note that the runtime is cubic in the number of response variables

verbose — print additional information

Examples

```
# Generate data
n = 100
p = 500
X = matrix(sample(0:2, n*p, replace=TRUE), nrow=n)
y = rnorm(n)
sex = as.factor(sample(1:2, n, replace=TRUE))

# Fit model for all markers
pValues = glmApply( y ~ sex + sex:SNP, features=X, terms=c(3,4))


# Multivariate model
n = 100
p = 1000
m = 10

Y = matrix(rnorm(n*m), nrow=n, ncol=m)
X = matrix(rnorm(n*p), nrow=n, ncol=p)

res = glmApply( Y ~ SNP, features = X, terms=2, multivariateTest=TRUE)

# p-values for univariate hypothesis test of each feature against
#  each response
res$pValues

# p-values for multivariate hypothesis test of each feature against
#  all responses are the same time
# returns the results of the Hotelling and Pillai tests
res$pValues_mv

# The multivariate test for X[,1]
res$pValues_mv[1,]

# The result is the same as the standard tests in R
fit = manova( Y ~ X[,1])

summary(fit, test="Hotelling-Lawley")
summary(fit, test="Pillai")
```

influence.lrgpr *Regression Diagnostics*

Description

Basic quantities for regression diagnostics from fit of `lrgpr`

Usage

```
influence.lrgpr(model, ...)
```

Arguments

`model`	model fit with `lrgpr`
`...`	other arguments

leverage.lrgpr *Regression Diagnostics*

Description

Basic quantities for regression diagnostics from fit of `lrgpr`

Usage

```
leverage.lrgpr(object)
```

Arguments

`object`	model fit with `lrgpr`

lm.influence.lrgpr *Regression Diagnostics*

Description

Basic quantities for regression diagnostics from fit of `lrgpr`

Usage

```
lm.influence.lrgpr(object, ...)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments

logLik.lrgpr *Extract Log-Likelihood*

Description

Log-Likelihood for model fit by `lrgpr`

Usage

```
logLik.lrgpr(object, ...)
```

Arguments

`object`	model fit with `lrgpr`
`...`	other arguments

loss.lrgpr *Loss function*

Description

Compare observed and fitted response under some loss function

Usage

```
loss.lrgpr(y, yhat, family)
```

Arguments

`y`	observed response
`yhat`	fitted response
`family`	"gaussian" or "binomial"

`lrgpr` *Fit a Low Rank Gaussian Process Regression (LRGPR) / Linear Mixed Model (LMM)*

Description

'lrgpr' is used to fit LRGPR/LMM models that account for covariance in response values, but where the scale of the covariance is unknown. Standard linear modeling syntax is used for the model specification in addition to a covariance matrix or its eigen-decomposition.

Usage

```
lrgpr(formula, decomp,
  rank = max(length(decomp$d), length(decomp$values)),
  delta = NULL, nthreads = detectCores(logical = TRUE),
  W_til = NULL, scale = TRUE)
```

Arguments

`formula`	standard linear modeling syntax as used in 'lm'
`decomp`	eigen-decomposition produced from eigen(K), where K is the covariance matrix. Or singular value decomposition svd(X[,1:100]) based on a subset of markers
`rank`	decomposition is truncated to the first rank eigen-vectors
`delta`	ratio of variance components governing the fit of the model. This should be estimated from a previous evaluation of 'lm' on the same response and eigen-decomposition
`nthreads`	number of threads to use for parallel execution
`W_til`	markers used to construct decomp that should now be removed from costruction of decomp. This is the proximal contamination term of Listgarten, et al. (2012)
`scale`	should W_til be scaled and centered

Value

`coefficients`	regression coefficients for each covariate
`p.values`	p-values from Wald test of each coefficient
`sd`	standard deviation of each coefficient estimate
`sigSq_e`	variance component σ_e^2 corresponding to the residual error
`sigSq_a`	variance component σ_a^2 corresponding the scale of the covariance, K

`delta`	ratio of variance components: σ_e^2/σ_a^2
`rank`	the rank of the random effect
`logLik`	log-likelihood of the model fit
`fitted.values`	estimated response values: y_hat
`alpha`	BLUP of the random effect
`Sigma`	variance-covariate matrix of estimate of beta
`hii`	diagonals of the matrix H such that y_hat = Hy
`y`	responses
`x`	design matrix
`df`	effective degrees of freedom: trace(H) based on Hoffman (2013)
`residuals`	residuals of model fit: y - y_hat
`AIC`	Akaike information criterion
`BIC`	Bayesian information criterion
`GCV`	generalized cross-validation
`eigenVectors`	eigen-vectors in decomp
`eigenValues`	eigen-values in decomp
`df.residual`	n - ncol(X)
`rank`	rank of decomposition used, where only non-negative eigen/singular values are considered

Details

'lrgpr' fits the model:

$$y = X\beta + \alpha + \epsilon$$

$$\alpha \sim N(0, K\sigma_a^2)$$

$$\epsilon \sim N(0, \sigma_e^2)$$

where

$$\delta = \sigma_e^2/\sigma_a^2$$

In practice the eigen-decomposition of K, and not K itself is required. The rank can be set to use only eigen-vectors 1:rank in the model.

This package allows hypothesis tests of single coefficients using fit$p.values which fits a Wald test. Composite hypothesis tests of multiple coefficients are performed with wald(fit, terms=1:3).

Note that likelihood ratio tests with linear mixed models do not perform well and the resulting p-values often do not follow a uniform distribution under the null (Pinheiro and Bates, 2000). We strongly advise against using it with this model.

'lrgpr' uses the algorithm of Lippert, et al. (2011).

See Hoffman (2013) for an interpretation of the linear mixed model.

References

Kang, H. M., et al. (2010) Variance component model to account for sample structure in genome-wide association studies. _Nature Genetics_ 42, 348-54

Lippert, C., et al. (2011) FaST linear mixed models for genome-wide association studies. _Nature Methods_ 9, 525-26

Listgarten, J., et al. (2012) Improved linear mixed models for genome-wide association studies. _Nature Methods_ 8, 833-5

Rasmussen, C. E. and Williams, C. K. I. (2006) Gaussian processes for machine learning. MIT Press

Pinheiro, J. C. and Bates, D. M. (2000) Mixed-Effects Models in S and S-PLUS. Springer, New York

Hoffman, G. E. (2013) Correcting for Population Structure and Kinship Using the Linear Mixed Model: Theory and Extensions. _PLoS ONE_ 8(10):e75707

See Also

'wald', 'lrgprApply'

Examples

```
# Generate random data
set.seed(1)
n <- 200
y <- rnorm(n)
K <- crossprod( matrix(rnorm(n*1000), ncol=n) )
age <- rpois(n, 50)
sex <- as.factor(sample(1:2, n, replace=TRUE))
decomp <- eigen(K)

# Fit the model
fit <- lrgpr( y ~ sex + age, decomp)

# Print results
fit

# Print more detailed results
summary(fit)

# P-values for each covariate
fit$p.values

# Visualize fit of the model like for 'lm'
par(mfrow=c(2,2))
plot(fit)

# Composite hypothesis test using Wald's test
# Joint test of coefficients 2:3
wald( fit, terms=2:3)
```

`lrgprApply` *Fit a Low Rank Gaussian Process Regression (LRGPR) / Linear Mixed Model (LMM) for many markers*

Description

'lrgprApply' is used to fit LRGPR/LMM models that account for covariance in response values, but where the scale of the covariance is unknown. It returns p-values equivalent to the results of lrgpr() and wald(), but is designed to analyze thousands of markers in a single function call.

Usage

```
lrgprApply(formula, features, decomp, terms = NULL,
  rank = max(length(decomp$d), length(decomp$values)),
  map = NULL, distance = NULL, dcmp_features = NULL,
  W_til = NULL, scale = TRUE, delta = NULL,
  reEstimateDelta = FALSE,
  nthreads = detectCores(logical = TRUE),
  verbose = FALSE)
```

Arguments

`formula`	standard linear modeling syntax as used in 'lm'. SNP is a place holder for the each successive column of features
`features`	a matrix where the statistical model is evaluated with SNP if formula replace by each column successively
`decomp`	eigen-decomposition produced from eigen(K), where K is the covariance matrix. Or singular value decomposition svd(features[,1:100]) based on a subset of markers
`terms`	indices of the coefficients to be tested. The indices corresponding to SNP are used if terms is not specified
`rank`	decomposition is truncated to the first rank eigen-vectors
`W_til`	markers used to construct decomp that should now be removed from costruction of decomp. This is the proximal contamination term of Listgarten, et al. (2012)
`scale`	should W_til be scaled and centered
`delta`	ratio of variance components governing the fit of the model. This should be estimated from a previous evaluation of 'lm' on the same response and eigen-decomposition
`reEstimateDelta`	should delta be re-estimated for every marker. Note: reEstimateDelta=TRUE is much slower
`nthreads`	number of to use for parallel execution
`verbose`	print extra information

Examples

```
# Generate data
n = 100
p = 500
X = matrix(sample(0:2, n*p, replace=TRUE), nrow=n)
y = rnorm(n)
sex = as.factor(sample(1:2, n, replace=TRUE))

K = tcrossprod(matrix(rnorm(n*n*3), nrow=n))
decomp = eigen(K, symmetric=TRUE)

# Fit null model
fit = lrgpr( y ~ sex, decomp)

# Fit model for all markers
pValues = lrgprApply( y ~ sex + sex:SNP, features=X, decomp, terms=c(3,4), delta=fit$delta)
```

---

plot.criterion.lrgpr

Plot AIC/BIC/GCV values for lrgpr() model as rank changes

Description

'plot.criterion.lrgpr' plots the criteria returned by 'criterion.lrgpr'

Usage

```
plot.criterion.lrgpr(x, col = rainbow(3), ...)
```

Arguments

x	list returned by 'criterion.lrgpr'
col	array of 3 colors
...	other arguments

See Also

criterion.lrgpr

plot.cv.lrgpr *Plot Results of Cross-validation*

Description

Plot results of 'cv.lrgpr', which fits cross-validation for multiple ranks of the LRGPR

Usage

```
plot.cv.lrgpr(x,
  ylim = c(min(x$cve - x$cvse), max(x$cve + x$cvse)),
  xlim = range(x$rank), pch = 20, col = "red",
  main = "Cross validation", xlab = "# of markers used",
  ylab = "Cross validation error", ...)
```

Arguments

x	result of cv.lrgpr
ylim	limits of y-axis
xlim	limits of x-axis
pch	pch
col	col
main	main
xlab	xlab
ylab	ylab
...	other parameters fed to plot()

plot.lrgpr *Plot Diagnostics for an* lrgpr *Object*

Description

Six plots (selectable by \"which\") are currently available: a plot of residuals against fitted values, a Scale-Location plot of sqrt(| residuals |) against fitted values, a Normal Q-Q plot, a plot of Cook's distances versus row labels, a plot of residuals against leverages, and a plot of Cook's distances against leverage/(1-leverage). By default, the first three and \"5\" are provided.

Usage

```
plot.lrgpr(x, which = c(1L:3L, 5L),
  caption = list("Residuals vs Fitted", "Normal Q-Q", "Scale-Location", "Cook's d
  panel = if (add.smooth) panel.smooth else points,
  sub.caption = NULL, main = "",
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  ..., id.n = 3, labels.id = names(residuals(x)),
  cex.id = 0.75, qqline = TRUE, cook.levels = c(0.5, 1),
  add.smooth = getOption("add.smooth"),
  label.pos = c(4, 2), cex.caption = 1)
```

Arguments

x	lrgpr object.
which	if a subset of the plots is required, specify a subset of the numbers \"1:6\".
caption	captions to appear above the plots; \"character\" vector or \"list\" of valid graphics annotations, see \"as.graphicsAnnot\". Can be set to \"""\" or \"NA\" to suppress all captions.
panel	panel function. The useful alternative to \"points\", \"panel.smooth\" can be chosen by \"add.smooth = TRUE\".
sub.caption	common title-above the figures if there are more than one; used as \"sub\" (s.\"title\") otherwise. If \"NULL\", as by default, a possible abbreviated version of \"deparse(x$call)\" is used.
main	title to each plot-in addition to \"caption\".
ask	logical; if \"TRUE\", the user is _ask_ed before each plot, see \"par(ask=.)\".
...	other parameters to be passed through to plotting functions.
id.n	number of points to be labelled in each plot, starting with the most extreme.
labels.id	vector of labels, from which the labels for extreme points will be chosen. \"NULL\" uses observation numbers.
cex.id	magnification of point labels.
qqline	logical indicating if a \"qqline()\" should be added to the normal Q-Q plot.
cook.levels	levels of Cook's distance at which to draw contours.
add.smooth	logical indicating if a smoother should be added to most plots; see also \"panel\" above.
label.pos	positioning of labels, for the left half and right half of the graph respectively, for plots 1-3.
cex.caption	controls the size of \"caption\".

See Also

plot.lm

predict.lrgpr *Predict response*

Description

Predict response values after training with `lrgpr`. Leaving X_test amd K_test as NULL returns the fitted values on the training set

Usage

```
predict.lrgpr(object, X_test = NULL, K_test = NULL, ...)
```

Arguments

`object`	model fit from `lrgpr` on training samples
`X_test`	design matrix of covariates for test samples
`K_test`	covariance matrix between samples in the test set and training set
`...`	other arguments

print.lrgpr *Print Values*

Description

Print details for fit from `lrgpr`

Usage

```
print.lrgpr(x, ...)
```

Arguments

`x`	model fit from `lrgpr`
`...`	other arguments

print.summary.lrgpr *Object Summaries*

Description

Print summary for fit from `lrgpr`

Usage

```
print.summary.lrgpr(x, ...)
```

Arguments

`x`	model fit from `lrgpr`
`...`	other arguments

read.fam *Read plink FAM/TFAM files*

Description

Read FAM/TFAM file into a dataframe. This function is the same as read.tfam

Usage

```
read.fam(file)
```

Arguments

`file`	location of FAM/TFAM file

read.tfam *Read plink FAM/TFAM files*

Description

Read FAM/TFAM file into a dataframe. This function is the same as read.fam

Usage

```
read.tfam(file)
```

Arguments

`file`	location of FAM/TFAM file

residuals.lrgpr *Extract Model Residuals*

Description

Residuals fitted with `lrgpr`

Usage

```
residuals.lrgpr(object, type = "working", ...)
```

Arguments

`object`	model fit with `lrgpr`
`type`	the type of residual, but there is only one option here
`...`	other arguments

rstandard.lrgpr *Regression Deletion Diagnostics*

Description

Basic quantities for regression deletion diagnostics from fit of `lrgpr`

Usage

```
rstandard.lrgpr(model, ...)
```

Arguments

`model`	model fit with `lrgpr`
`...`	other arguments

set_missing_to_mean *Replace Missing Values with Mean*

Description

For each column, replace NA values with the column mean

Usage

```
set_missing_to_mean(A)
```

Arguments

A	matrix

summary.lrgpr *Summarizing LRGPR / Linear Mixed Model Fits*

Description

Print summary for fit from lrgpr

Usage

```
summary.lrgpr(object, ...)
```

Arguments

object	model fit from lrgpr
...	other arguments

vcov.lrgpr *Calculate Variance-Covariance Matrix for a lrgpr Object*

Description

Returns the variance-covariance matrix of the main parameters of a fitted model object

Usage

```
vcov.lrgpr(object, ...)
```

Arguments

object	model fit with lrgpr
...	other arguments

`wald` *Composite hypothesis test of multiple coefficients*

Description

'wald' performs a multi-dimensional Wald test against H0: beta_i...beta_j = 0 using the estimated coefficients and their variance-covariance matrix

Usage

```
wald(fit, terms)
```

Arguments

`fit`	result of fitting with 'lrgpr'
`terms`	indices of the coefficients to be tested

Details

The Wald statistic is

$$\beta_h^T \Sigma_h^{-1} \beta_h \sim \chi^2_{|h|}$$

where

$$h$$

specifies the coefficients being tested and

$$h$$

is the number of entries

See Also

'lrgpr'

Index